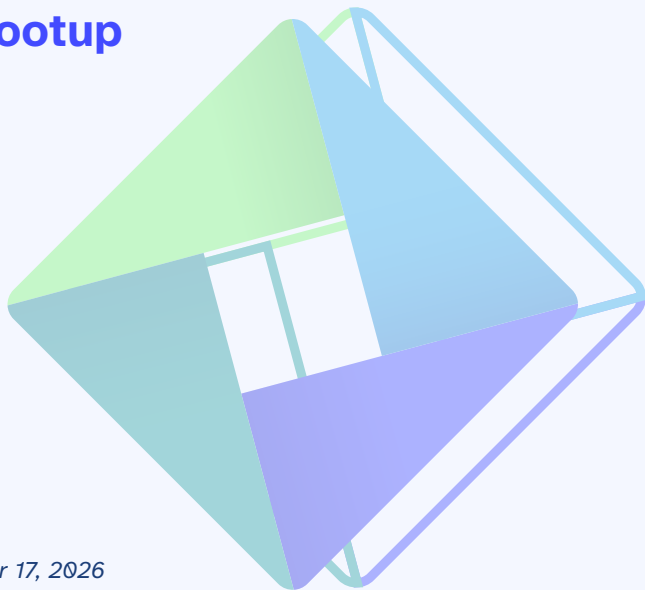


Meeting Early-Bootup Requirements

Orchestrating Zephyr and Linux
in AMP Systems



Stefan Kratochwil

Zephyr Meetup Zürich - September 17, 2026

Agenda

- Asymmetric Multiprocessing - An Overview
- The Hardware Platform
- Zephyr on the M7
- OpenAMP Basics
- Signalling and Data Exchange
- Integration and Result

What is AMP?

- Multiple **non-interchangeable** cores on one SoC
 - Each core runs its **own software context**
(individual OS instances, different OSs, or bare metal)
 - Statically assigned roles, (mostly) private memory, explicit IPC
- In contrast, **SMP**: one kernel, all cores are treated equal, threads migrate freely
- The asymmetry reflects in the *software*, not necessarily in the hardware
 - Different cores are typical for AMP, but they do not *define* AMP
 - big.LITTLE under one Linux scheduler is still **SMP**

Usage Scenario

Everything started with a few requirements:

1. Linux on application core
2. Zephyr on realtime core
3. Zephyr must be operational asap
4. Target SoC is a NXP i.MX8MP



Figure 1: Toradex Verdin iMX8M Plus

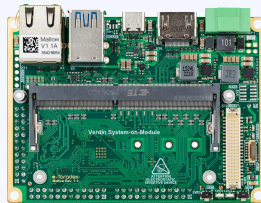


Figure 2: Toradex Mallow Carrier Board

Let's have a look

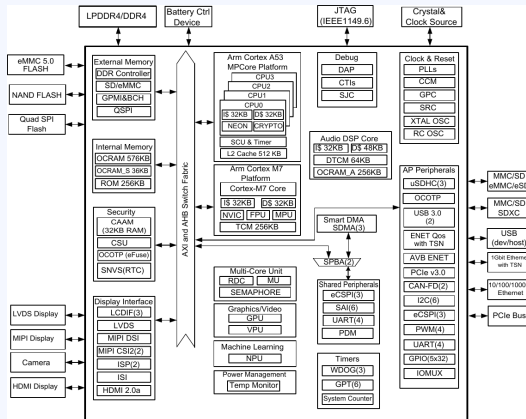


Figure 3: Block diagram of the i.MX8MP. Reference Manual, Rev. 3, 08/2024.

Let's have a look - The two platforms of interest

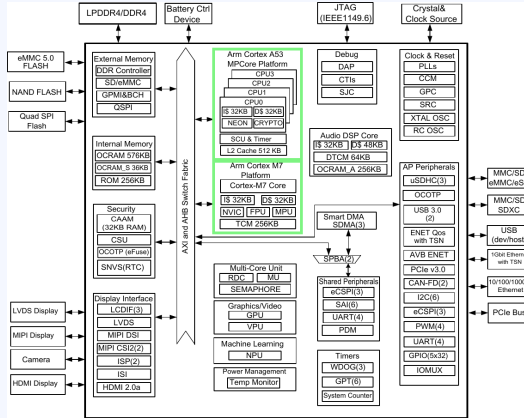


Figure 4: Cores, caches and TCMs are grouped into *platforms*.

Platform starting order

A look into the hardware manual reveals:

IOMUXC_GPR_GPR22 field descriptions					
Field	Description				
31-16 -	This field is reserved.				
15-4 -	This field is reserved.				
3 GPR_M7_HCLK_GATE_EN	Gate off CM7 hclk 0 gate on 1 gate off				
2 GPR_M7_HCLK_AUTO_GATE_EN	Gate CM7 hclk automatically 0 depends on the value of GPR_M7_HCLK_GATE_EN 1 ignore the value of GPR_M7_HCLK_GATE_EN				
1 -	This field is reserved.				
0 GPR_M7_CPUWAIT	CM7 in wait mode 0 do not let CM7 enter wait mode 1 let CM7 enter wait mode				
3034_0058	General Purpose Register 22 (IOMUXC_GPR_GPR22)	32	R/W	0000_0001h	8.2.3.23/ 1391

Figure 5: On the i.MX8MP, the M7 is held in reset by default.

→ **M7** remains in reset until **started from elsewhere**

Obvious next step: Check Zephyr Documentation

- Good news: the board *is* supported
- It is "Not actively maintained"
- But: It's still listed on the Zephyr website
→ Seems to pass the CI on github
- And: There is an example on how to run the M7 from u-boot!



```
ext4load mmc 2:2 ${loadaddr} /<path-to-binary>/zephyr.bin
cp.b ${loadaddr} 0x7e0000 <size_of_binary_in_bytes>
bootaux 0x7e0000
```

Listing 1: Loading and starting the M7 from u-boot

A quick test: We have a Zephyr shell

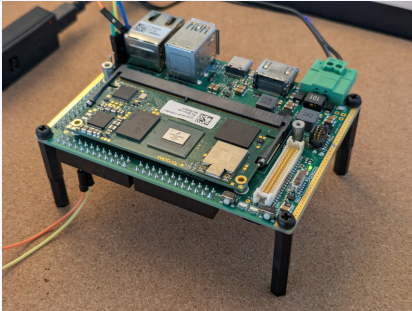


Figure 6: Hardware setup with UART 1 and 3 connected.

```
stefan@KALMSKR ~ % tio /dev/serial/by-id/usb-FTDI_FT232R_USB_UART_A9QXTBFJ-if00-port0
[08:54:24.626] tio 62083fd
[08:54:24.626] Press ctrl-t q to quit
[08:54:24.808] Connected to /dev/serial/by-id/usb-FTDI_FT232R_USB_UART_A9QXTBFJ-if00-port0
*** Booting Zephyr OS build v4.4.0 ***
[00:00:00.000,000] <inf> openamp_rsc_table: Starting application threads!
[00:00:00.000,000] <inf> openamp_rsc_table: OpenAMP[remote] Linux responder demo started

uart:--$ [00:00:04.902,000] <inf> openamp_rsc_table: OpenAMP[remote] Linux sample client responder started
[00:00:04.902,000] <inf> openamp_rsc_table: OpenAMP[remote] Linux TTY responder started
[00:00:04.902,000] <inf> openamp_rsc_table: OpenAMP[remote] Linux sample client responder started
[00:00:04.902,000] <inf> openamp_rsc_table: OpenAMP[remote] Linux TTY responder started
uart:--$
```

Figure 7: Zephyr shell, grabbed from UART 1.

→ Zephyr compiles and works as expected

What is OpenAMP

- Protocol stack for AMP systems (from top to bottom):
 - `rpmsg` → service announcements / endpoints
 - `virtio` → interprocess communication
 - `libmetal` → hardware abstraction
- Mainly built for MMU-less platforms
- Implements both possible roles (*master* or *slave*)

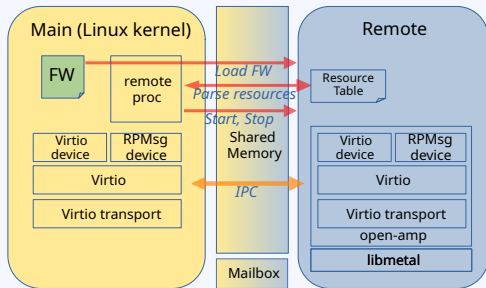


Figure 8: Here Linux loads the image (FW) to the remote.

libmetal

- Provides a common API for system-level operations:
 - Registering interrupt handlers
 - Accessing shared memory when a MMU is present
 - Cross-platform locking mechanisms
- Foundation of OpenAMP
 - Keeps OpenAMP fully hardware-agnostic

virtio

- Standardized interface for *virtual* devices
- Appear as regular peripherals to the host
- Supported transports:
 - PCI Bus (most common, device appears as PCI(e) device)
 - **MMIO** (used when no PCI is available, device interface is memory-mapped)
 - Channel I/O (used when neither PCI nor MMIO is available, e.g. S/390)
- Supports a broad range of device types:
 - Network device
 - **Console device**
 - **rpmsg**
 - Input device
 - ...

rpmsg

- Messaging bus implemented on top of virtio
- On Zephyr: used for dynamic name service announcements

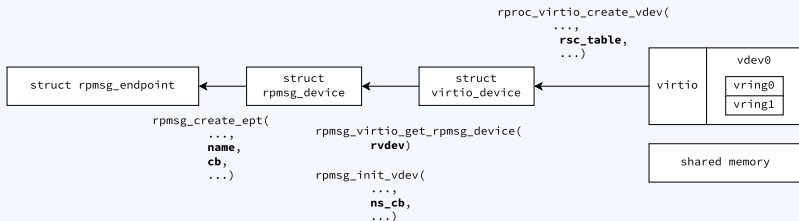


Figure 9: Object chain built during rpmsg initialization.

Enabling OpenAMP in Zephyr

- Comes as Zephyr module
- Requires Zephyr-specific libmetal fork (also comes as module)
- Very cool: Zephyr brings samples (more on the next slide)

```
manifest:
  projects:
    - name: zephyr
      import:
        name-allowlist:
          - libmetal
          - open-amp
```

Listing 2: west manifest excerpt

```
CONFIG_OPENAMP=y
CONFIG_OPENAMP_MASTER=n
CONFIG_OPENAMP_RSC_TABLE=y
# . . .
```

Listing 3: related Kconfig symbols

Interlude: How do Linux and Zephyr find each other?

- **Resource Table**
- Defined by Linux's remoteproc framework
- Implemented by the remote device
- Typically **placed in special ELF section**
- Goal:
 - Remote device shall tell Linux what it needs
 - Linux's driver reports back when ready

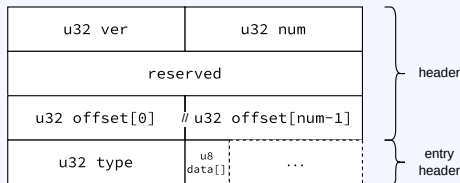


Figure 10: Layout of the resource table

How Zephyr populates the resource entries

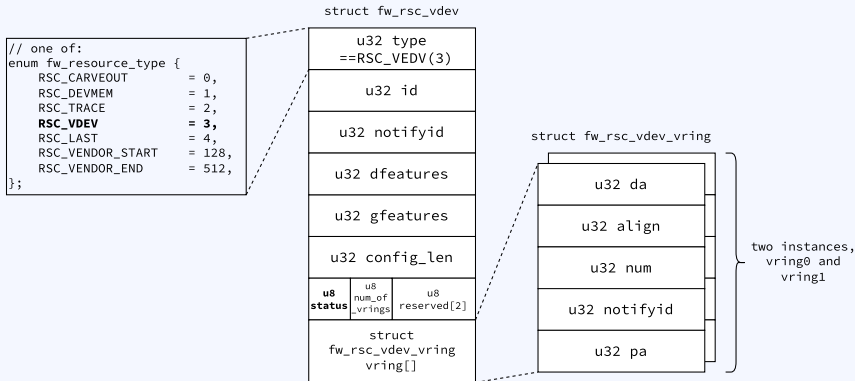


Figure 11: One vring for each direction. If RAM_CONSOLE is used: additional RSC_TRACE.

Handshake via Resource Table

- First of all: **CONFIG_OPENAMP_COPY_RSC_TABLE=y**
- Zephyr and Linux rendezvous via the status field
- Zephyr
 - first initializes an rpmsg endpoint, then
 - waits by polling `rproc_virtio_get_status()` until
 - the `VIRTIO_CONFIG_STATUS_DRIVER_OK` bit is set
- Linux
 - probe()s the `rpmsg_virtio` driver with `virtio_dev_probe()`, which
 - calls `virtio_add_status(dev, VIRTIO_CONFIG_S_DRIVER_OK)`

Important

Only works if resource table is in shared memory before Linux driver is loaded

Load model vs. attach model

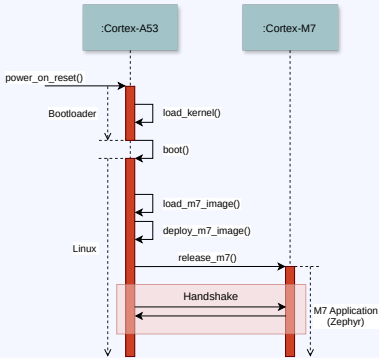


Figure 12: Load model: M7 started by Linux

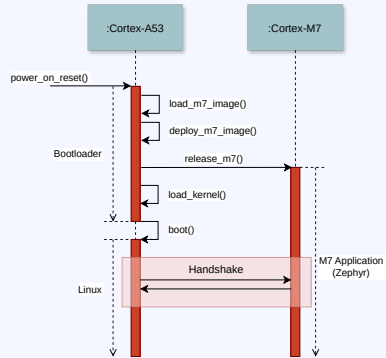


Figure 13: Attach model: M7 started by u-boot

Zephyr samples for OpenAMP

- `samples/subsys/ipc/openamp_rsc_table`
- A lot of board definitions, also for the i.MX8MP EVK
- But: not for the Toradex Verdin
- And: subsystem documentation only describes the *load model*
- However: we can salvage board configs and devicetrees

Inter-processor Interrupts

- Both platforms need a way to interrupt each other
- General options:
 - Instruction set feature
 - Hardware FIFOs / Mailboxes
 - Hardware Semaphores
 - IPI Facilities
- On i.MX8MP: **Messaging Unit (MU)**
 - Two sides, A (A53) and B (M7)
 - Four independent channels
 - One Pair of TX/RX registers per channel
 - 32 bit per channel

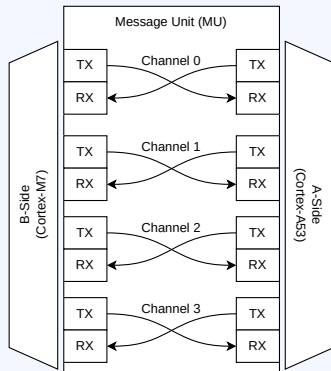


Figure 14: Simplified layout of the Messaging Unit.

Message Unit: Zephyr and Linux device tree excerpts

```
&mailbox0 {  
    status = "okay";  
};
```

Listing 4: On Zephyr, MU is active by default

```
imx8mp_cm7: imx8mp-cm7 {  
    compatible = "fsl,imx8mp-cm7";  
    clocks = <&clk IMX8MP_CLK_M7_CORE>;  
    fsl,auto-boot;  
    mbox-names = "tx", "rx", "rxdb";  
    /* MU channels, encoded as <&mu type index>:  
    * type 0=tx, 1=rx, 3=rxdb; index = MU register num  
    * tx    <&mu 0 1>  Linux -> Zephyr via channel 1  
    * rx    <&mu 1 0>  Linux <- Zephyr via channel 0  
    * rxdb  <&mu 3 1>  rx doorbell on channel 1 */  
    mbox = <&mu 0 1>, <&mu 1 0>, <&mu 3 1>;  
    /* We need these later, but they belong here: */  
    memory-region = <&vdev0buffer>, <&vdev0vring0>,  
        <&vdev0vring1>, <&rsc_table>;  
}
```

Listing 5: Linux side

Linux has expectations...

- Linux **always transmits** on **channel 0**
- Linux **always receives** from **channel 1**
- This is hardcoded in `remoteproc_core.c#rproc_alloc_vring()`
→ Not only applicable to i.MX8, but general behavior!
- This means: The Zephyr side needs to adapt!

How Zephyr's openamp_rsc_table sample handles this

```
static void platform_ipm_callback(  
    const struct device *dev,  
    void *context, uint32_t id,  
    volatile void *data)  
{  
    LOG_DBG("%s:_msg_received_from_mb_%d",  
            __func__, id);  
    k_sem_give(&data_sem);  
}
```

Listing 6: RX path: the sample accepts kicks from *any* MU channel

```
CONFIG_OPENAMP_RSC_TABLE_IPM_RX_ID=0  
CONFIG_OPENAMP_RSC_TABLE_IPM_TX_ID=1
```

Listing 7: TX path: required Kconfig symbols

Note

The i.MX8MP EVK defconfig has it the other way round: RX_ID=1, TX_ID=0

Some more important Kconfig for Zephyr

```
CONFIG_IPM_IMX_MAX_DATA_SIZE_4=y  
CONFIG_IPM_IMX_MAX_DATA_SIZE_16=n
```

Listing 8: Linux expects a kick on *one* mailbox with a 32 bit value.

Shared memory carveout

- Shared memory is accessed by two different architectures (aarch64 and armv7e-m)
- Memory layout represents a *contract* both sides need to agree on
- System device tree? Not ready yet
- For now: manually maintain devicetrees for both sides

Shared memory layout

Borrowed from other samples:

Start address	Size	Name	Filled by	Purpose
0x55000000	0x8000	vdev0vring0	Linux	M7 → A53 queue
0x55008000	0x8000	vdev0vring1	Linux	A53 → M7 queue
0x55010000	0x100000	vdev0buffer	Linux	rpmsg buffer pool (DMA pool)
0x55110000	0x1000	rsc-table	Zephyr	publishes its resource table here

Zephyr and Linux device tree excerpts

```
shram: memory@55000000 {  
    compatible = "mmio-sram";  
    reg = <0x55000000 0x110000>;  
};  
  
rsc_table: memory@55110000 {  
    compatible = "mmio-sram";  
    reg = <0x55110000 0x1000>;  
};
```

Listing 9: Coarse grained regions on Zephyr

```
vdev0vring0: vdev0vring0@55000000 {  
    reg = <0 0x55000000 0 0x8000>; no-map;  
};  
  
vdev0vring1: vdev0vring1@55008000 {  
    reg = <0 0x55008000 0 0x8000>; no-map;  
};  
  
vdev0buffer: vdev0buffer@55010000 {  
    compatible = "shared-dma-pool";  
    reg = <0 0x55010000 0 0x100000>; no-map;  
};  
  
rsc_table: rsc-table@55110000 {  
    reg = <0 0x55110000 0 0x1000>; no-map;  
};
```

Listing 10: Fine grained "/reserved-memory" on Linux

Integrating with u-boot

- Zephyr image must be at well known location
- Example: /lib/firmware on rootfs
- Insert load comand into u-boot environment and make it execute automatically

```
m7_image=/lib/firmware/m7/zephyr.bin
m7_itcm_addr=0x7e0000
m7_itcm_size=0x20000
m7_boot=if ext4load ${mender_uboot_root} ${loadaddr} ${m7_image}; then
    if test 0x${filesize} -gt ${m7_itcm_size}; then
        echo M7: ${m7_image} too big for ITCM: 0x${filesize} bytes max ${m7_itcm_size};
    else
        cp.b ${loadaddr} ${m7_itcm_addr} ${filesize};
        dcache flush; bootaux ${m7_itcm_addr};
        setenv tdxargs ${tdxargs} clk_imx8mp.mcore_booted=1;
    fi;
else echo M7: failed to load ${m7_image}; fi
```

Listing 11: An example for a load command in the u-boot environment

The result

Demo

Thank you!



Stefan Kratochwil
Embedded Software
Engineer

stefan.kratochwil@inovex.de

+49 172 / 98 17 860

Ludwig-Erhard-Allee 6
76131 Karlsruhe

inovex in an IT project center driven by innovation and quality, focusing its services on 'Digital Transformation'.

- founded 1999
- 500+ employees
- 7 offices across germany
- 1 office in switzerland



www.inovex.de

Backup Slides

Backup: Bootup of u-boot and Zephyr

<pre>U-Boot 2024.07-0+git.3f772959581c (Jul 01 2024 - 18:07:18 +0000) DDR configured as dual rank SEC0: RNG instantiated Normal Boot WDT: Started watchdog@30230000 with servicing every 1000ms (60s timeout) Trying to boot from BOOTROM Boot Stage: Primary boot Find img info 0x4002f000, size 1048 Need continue download 1024 U-Boot 2024.07-0+git.3f772959581c (Jul 01 2024 - 18:07:18 +0000) CPU: Freescale i.MX3MP[8] rev1.1 1600 MHz (running at 1200 MHz) CPU: Industrial temperature grade (-40C to 105C) at 42C Reset cause: POR DRAM: 8 GiB Core: 177 devices, 31 uclasses, devicetree: separate WDT: Started watchdog@30230000 with servicing every 1000ms (60s timeout) MMC: FSL_SDHC: 1, FSL_SDHC: 2 Loading Environment from MMC... Reading from MMC(2)... OK MISSING TORADAX CONFIG BLOCK In: serial@30830000 Out: serial@30830000 Err: serial@30830000 Model: Toradex Verdin iMX3M Plus WB on Verdin Development Board SEC0: RNG instantiated Net: pca953x gpio-expander@21: Error reading output register eth1: ethernet@30b00000, eth0: ethernet@30b00000 [PRIME] 66744 bytes read in 3 ms (21.2 MiB/s) ## No elf image at address 0x007e0000 ## Starting auxiliary core stack = 0x20004DC0, pc = 0x00005515... Hit any key to stop autoboot: 1 [0] 0:~\$</pre>	<pre>stefan@KALMSKR /tmp/terminal_recording % tio /dev/serial/by-id/usb-FTDI_FT232R_USB_UART_A9QXTBFJ-if00-port0 [11:16:29.601] tio 62083fd [11:16:29.601] Press ctrl-t q to quit [11:16:29.730] Connected to /dev/serial/by-id/usb-FTDI_FT232R_USB_UART_A9QXTBFJ-if00-port0 *** Booting Zephyr OS build v4.4.0 *** [00:00:00.000,000] <inf> openamp_rsc_table: Starting application threads! [00:00:00.000,000] <inf> openamp_rsc_table: OpenAMP[remote] Linux response r demo started uart:-\$</pre>
--	--

Figure 15: Two serial connections, one to each platform. Left: u-boot loading and starting the M7. Right: Zephyr shell coming up before Linux kernel boots.

Backup: Zephyr shell over RPMsg

<pre>[OK] Reached target System Time Set. [OK] Reached target Timer Units. [OK] Listening on D-Bus System Message Bus Socket. Starting sshd.socket... [OK] Listening on sshd.socket. [OK] Reached target Socket Units. [OK] Reached target Basic System. [OK] Started Kernel Logging Service. [OK] Started System Logging Service. [OK] Started Runs mender commit on each.atteer test to validate the syst ess. Starting D-Bus System Message Bus... Starting Network update service (proof-of-concept)... [OK] Started Serial Getty on ttyxc1. [OK] Started Serial Getty on ttyxc2. [OK] Reached target Login Proepts. Starting User Login Management... Starting OpenSSH Key Generation... [OK] Started D-Bus System Message Bus. [OK] Finished OpenSSH Key Generation. [OK] Started User Login Management. inovex Distro 1.0.0 inovex-imx8mp ttyxc2 inovex-imx8mp login: root root@inovex-imx8mp:~# root@inovex-imx8mp:~# root@inovex-imx8mp:~# microcom /dev/ttyRPMMSG0 [00:00:04.893,000] <inf> openamp_rsc_table: OpenAMP[remote] Linux sample cl ient responder started [00:00:04.893,000] <inf> openamp_rsc_table: OpenAMP[remote] Linux TTY respo nder started ipcs:~\$ [0] 0:~\$</pre>	<pre>uart:~\$ help Please press the <Tab> button to see all available commands. You can also use the <Tab> button to prompt or auto-complete all commands or its subcommands. You can try to call commands with <-h> or <--help> parameter for more info rmation. Shell supports following meta-keys: Ctrl + (a key from: abcdefklintpw) Alt + (a key from: bf) Please refer to shell documentation for more details. Available commands: clear : Clear screen. date : Date commands device : Device commands devmem : Read/write physical memory Usage: Read memory at address with optional width: devmem <address> [<width>] Write memory at address with mandatory width and value: devmem <address> <width> <value> help : Prints the help message. history : Command history. kernel : Kernel commands rm : Ignore lines beginning with 'rm ' resize : Console gets terminal screen size or assumes default in case the : readout fails. It must be executed after each terminal width change : to ensure correct text display. retval : Print return value of most recent command shell : Useful, not Unix-like shell commands. uart:~\$</pre>
---	---

Figure 16: Left: Fully functional Zephyr shell reachable from Linux at /dev/ttyRPMMSG0. Right: the Zephyr shell on the serial terminal.